CLAIMS

What is claimed:

1. A computer-implemented method for instrumentation of selected functions in an executable program, the selected functions initially occupying an original address space of the executable program, comprising:

generating instrumented versions of selected functions in relocation address space during program execution;

when a function is called by an instrumented version of a selected function within the relocation address space resulting in a first return-pointer value in the relocation address space, identifying a location in the original address space corresponding to the first return-pointer value as an original return-pointer value, associating the first return-pointer value with the original return-pointer value, substituting references to the original return-pointer value for references to the first return-pointer value, and replacing an instruction at the address indicated by the original return-pointer value with a breakpoint; and

when the breakpoint is encountered upon return of control at the original return-pointer value, obtaining the first return-pointer value associated with the original return-pointer value, and transferring control to an instruction at the address referenced by the first return-pointer value.

2. The method of claim 1, further comprising identifying RP-sensitive functions as the selected functions, wherein RP-sensitive functions are those functions that require a return pointer value in the original address space of the executable program.

3. The method of claim 2, further comprising:

patching entry points of the RP-sensitive functions with RP-entry breakpoints; and

generating an instrumented version of an RP-sensitive function upon encountering the RP-entry breakpoint at the entry point of the RP-sensitive function.

4. The method of claim 3, wherein each instrumented version of an RP-sensitive function has a corresponding original version function in the original address space, further comprising replacing instructions at entry points of the instrumented versions of the RP-sensitive functions with branch instructions targeting corresponding instrumented versions of the RP-sensitive functions.

5. The method of claim 4, further comprising identifying the RP-sensitive functions through analysis of code segments within the executable program.

6. The method of claim 4, further comprising identifying the RP-sensitive functions through an input list of identifier codes associated with RP-sensitive functions.

7. The method of claim 2, wherein each instrumented version of an RP-sensitive function has a corresponding original version function in the original address space, further comprising replacing instructions at entry points of the instrumented versions of the RP-sensitive functions with branch instructions targeting corresponding instrumented versions of the RP-sensitive functions.

8. The method of claim 2, further comprising identifying the RP-sensitive functions through analysis of code segments within the executable program.

9. The method of claim 2, further comprising identifying the RP-sensitive functions through an input list of identifier codes associated with RP-sensitive functions.

10. The method of claim 2, further comprising:

generating the relocation address space;

inserting RP-entry breakpoints at entry points of the RP-sensitive functions; and

upon encountering an RP-entry breakpoint during execution of the executable program, generating an instrumented version of the RP-sensitive function associated with the RP-entry breakpoint, and replacing the RP-entry breakpoint with a branch instruction that targets the instrumented version of the RP-sensitive function.

11. A computer-implemented method for instrumentation of selected functions in an executable program, the selected functions initially occupying an original address space of the executable program, comprising:

generating relocation address space;

identifying RP-sensitive functions in the executable program, wherein RP-sensitive functions are those functions that require a return pointer value in the original address space;

inserting RP-entry breakpoints at entry points of the RP-sensitive functions;

upon encountering an RP-entry breakpoint during execution of the executable program, generating an instrumented version of the RP-sensitive function associated with the RP-entry breakpoint, and replacing the entry point of the RP-sensitive function in the original address space with a branch instruction that targets the instrumented version of the RP-sensitive function;

when an instrumented version of RP-sensitive function is called from a function in the relocation address space whereby a return-pointer register stores a first return-pointer value within the relocation address space, identifying a location in the original address space corresponding to the first return-pointer value as an original return-pointer value, associating the first return-pointer value with the original return-pointer value, storing the original return-pointer value in the return-pointer register, and replacing an instruction at the address indicated by the original return-pointer value with an RP-return breakpoint; and

when the RP-return breakpoint is encountered upon return of control at the original return-pointer value, obtaining the first return-pointer value associated with the original return-pointer value, restoring the first return-pointer value to the return-pointer register, and transferring control via the return pointer register.

12. The method of claim 11, further comprising identifying the RP-sensitive functions through analysis of code segments within the executable program.

13. The method of claim 11, further comprising identifying the RP-sensitive functions through an input list of identifier codes associated with RP-sensitive functions.

14. An apparatus for instrumentation of selected functions in an executable program, the selected functions initially occupying an original address space of the executable program, comprising:

means for generating instrumented versions of selected functions in relocation address space during program execution;

means, responsive to a call to an instrumented version of a selected function from within the relocation address space whereby a first return-pointer value is within the relocation address space, for identifying a location in the original address space corresponding to the first return-pointer value as an original return-pointer value, associating the first return-pointer value with the original return-pointer value, substituting references to the original return-pointer value for references to the first return-pointer value, and replacing an instruction at the address indicated by the original return-pointer value with a breakpoint; and

means, responsive to encountering the breakpoint upon return of control at the original return-pointer value, for obtaining the first return-pointer value associated with the original return-pointer value, and transferring control to an instruction at the address referenced by the first return-pointer value.